Santa Clara High Technology Law Journal

Volume 7 | Issue 2 Article 2

January 1991

Theories of Computer Program Patentability

Jerome T. Tao

COMMENTS

THEORIES OF COMPUTER PROGRAM PATENTABILITY

Jerome T. Tao†

Much attention has been paid during the last twenty years to the possibility of obtaining patent protection for computer programs. Part of this attention has been due to the recent explosion in microprocessor and computer technology and the accompanying increase in the importance of the methods by which such computers are programmed.[1]

Part of the debate, however, results from conceptual difficulties inherent in the unique nature of such programs. Computer programs have been said to "defy the conceptual molds provided by the traditional patent and copyright systems."[2] A computer program can be described as a process, as an apparatus, or as something in between.

Some commentators see this difficulty as a reason to deny patent protection for all computer programs.[3] These definitional problems, however, should not be sufficient to provide a blanket reason for denying such protection to novel, useful and nonobvious computer programs which do not fall under any particular judicial exception to the patent laws.

This article will discuss some of the various ways of defining a computer "program," and the various ways in which each definition may be pigeonholed into the patent system, and explore the patenta-

† B.S.E.E., Cornell University, J.D. candidate, George Washington National Law Center. The author would like to thank Professor James Chandler for his invaluable assistance.

1. *See, e.g.*, Note, *The Policy Implications of Granting Patent Protection To Computer Software: An Economic Analysis*, 37 VAND. L. REV. 147, 148 nn.1 & 2 (1984).

2. Samuelson, Benson *Revisited: The Case Against Patent Protection For Algorithms and Other Computer Program-Related Inventions*, 39 EMORY L.J. 1025, 1128-1129 (1990). *See also* Bender, *Software Protection: The 1985 Perspective*, 7 W. NEW ENG. L. REV. 405, 407 (1985) ("In [certain] respects . . . the program is unusual, if not unique").

3. Newell, *Response: The Models Are Broken, The Models Are Broken!*, 47 U. PITT. L. REV. 1023 (1986). Professor Newell's article is a response to Chisum, *The Patentability of Algorithms*, 47 U. PITT. L. REV. 959 (1986).

bility problems raised by each definition. Before discussing the particularized issues involved in the patenting of computer programs, a brief overview of the requirements for patent protection and the basics of computer programming are provided.

I. GENERAL REQUIREMENTS FOR PATENT PROTECTION

The general patent statute is codified at 35 U.S.C. § 100 et seq. A patent may be obtained on "any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof. . . ."[4] The statute defines a patentable "process" as a "process, art or method, and includes a new use of a known process, machine, manufacture, composition of matter, or material."[5]

Once an invention has been determined to fall within one of these statutory classes of patentable subject matter, it must then meet the requirements of novelty, utility, and nonobviousness.[6] Additionally, the patent application itself must adequately set forth and describe the invention in terms such that other inventors can make and use the invention described therein.[7]

If an invention is deemed by the patent examiner to satisfy all of these requirements, a patent on the invention will be issued. The patent grants to the inventor the right to foreclose others from making, using or selling the patented invention for a nonrenewable term of 17 years from the date of issue of the patent.[8] Upon expiration of this statutory period, the subject matter of the patent passes into the public domain and may be freely used in any way by any party.

Obtaining a patent on an invention is a long, complex process, often spanning several years. The above discussion is necessarily only a broad overview; the particular requirements are beyond the scope of this article.

II. HOW A COMPUTER IS PROGRAMMED

Before exploring the definition of a computer "program," it is convenient to examine how a computer is programmed.

Broadly speaking, a computer operates according to sets of in-

4. 35 U.S.C. § 101 (1988).

5. 35 U.S.C. § 100 (1988). For a history of the statutory definitions, see Diamond v. Diehr, 450 U.S. 175, 181-84 (1981).

6. 35 U.S.C. §§ 102, 103 (1988). These requirements will not be discussed in detail here. For a more detailed discussion, see Graham v. John Deere Co., 383 U.S. 1 (1966).

7. 35 U.S.C. § 112 (1988). These are known as the "enablement" requirements.

8. 35 U.S.C. § 154 (1988).

structions designed by human operators to perform specific functions or to solve particular problems. These instructions are of three general types: those that direct the computer to receive specified data information (input instructions), those that direct the computer to manipulate the received data in certain ways (logical or operational instructions), and those that direct the computer to send certain data information to "peripheral" devices (output instructions).[9] These sets of input, operational and output instructions grouped together to direct the computer to perform a particular task are commonly referred to collectively as a computer "program."[10]

A program is thus a sequence of specific instructions, each of which directs the computer to perform certain internal arithmetic operations. These instructions can in turn be considered at two basic levels. The first is the low level "object code" which the computer recognizes and actually utilizes in its internal operation. Because modern computers operate internally according to binary logic, "machine code" consists of combinations of 1s and 0s.[11] In actuality, these 1s and 0s are stored electronically within the computer at discrete voltage levels (generally +5 volts for the "high" bits or 1s, and zero volts for the "low" bits or 0s). The second is the high-level "source code," which generally consists of instructions written in one of many computer "languages." These languages range in complexity, from very high-level languages whose instructions closely resemble the English language, to less high-level languages whose instructions are more technically precise and more accurately reflect the actual logical operations involved.[12] These high-level languages are merely programming aids which enable a human designer to conveniently map out the operations that he or

9. These "peripheral" devices can consist of conventional display units, data storage devices, and the like; however, as defined here, they may also consist of any electromechanical devices whose operation may be controlled by the computer, such as robotic arms, automated steel presses, etc. Hence, the term "output" signal is also intended to encompass what are also known as "control" signals.

10. Programs can be further categorized; for example, see Note, *supra* note 1, at 153, which categorizes computer programs as either "operating systems programs" or "applications programs." For this discussion, additional subclassifications such as these will all be grouped together generally as merely different types of computer programs.

11. These combinations typically occur in groups of eight digits or "bits," though modern processors often accommodate longer bit combinations or "words."

12. Assembly language, which consists of abbreviated symbols and base-16 numerical representations, is often considered a "mid"-level language because it is a fairly close representation of object code. Assembly language instructions are directly translatable into machine code on a one-to-one basis. For the purposes of this essay, however, and for ease of discussion, anything which is not binary object code will be considered high-level.

she wants the computer to perform. The computer cannot recognize the language instructions as such; they must be converted into object code for actual storage and execution. A high-level instruction may encompass several individual object code words.[13] Both high-level source code and low-level object code fall under the label of computer "coding."

The actual code instructions selected to direct a computer to perform a set of operations clearly cannot be the subject of a patent, because this would in effect protect the language or the particular instruction itself.[14] What a programmer may be interested in protecting, however, is the particular sequential *combination* of language instructions selected to perform a particular task.

Typically, however, different combinations of code instructions may be utilized to perform precisely the same operation. Thus, protecting a particular combination of steps in a given computer language would provide only limited protection. Broader protection could be gained if one could patent the underlying logical process which the particular language steps have been chosen to represent. In the computer programming world, this logical process is commonly described on paper by a flowchart.[15]

III. THEORIES OF PATENTABILITY

The debate over patentability has principally concerned whether a computer program, as such, falls within the definition of a patentable "process" set forth in 35 U.S.C. §§ 100 and 101. The answer to this question, of course, depends upon which of the levels discussed above one seeks to protect with a patent. There are three possible ways of defining computer programs so that they may fall within the patent laws. Each of them gives rise to distinct patentability problems.

A. *Patenting a program as a process.*

One may seek to patent a computer program as a process for accomplishing what the computer program is designed to do.

13. The higher the language level, the less representative it is of the number of actual bit sequences stored by the computer. See *supra* note 12.

14. *I.e.*, one could not patent the instruction "READ" or the code instruction "01101011."

15. Commentators have identified other, more abstract levels at which the operations of a computer may be analyzed. *See, e.g.*, Bender, *Software Protection: The 1985 Perspective*, 7 W. NEW ENG. L. REV. 405, 409 (1985) (discussing additionally the "architecture" stage and the "problem definition" stage). However, these additional levels tend to be less interesting analytically for patentability purposes.

Under this definition, each instruction step, or each group of related instructions, is represented as a claimed process step. At this level, protection is obtained essentially on the source or object code sequence used to implement the program.[16]

The protection obtained under this theory is rather limited. A patent holder would only be able to foreclose others from using a particular instruction sequence or its equivalent (i.e., the same sequence of steps written in a different computer language).[17] Because of the limited scope of protection, this theory raises few analytical problems.

B. *Patenting a program as the underlying "art."*

Alternatively, a computer program may be patented as the functional embodiment of the underlying "art." That is, the patent owner seeks to protect the logical or theoretical process which the particular code steps are intended to embody in functional form. This underlying process has been called the "computer program algorithm,"[18] and that terminology will be adopted here.

Conceptually, there are several important things to note about granting patent protection for this type of logical process. At this level, it is irrelevant which programming language is used to implement the logical process; *any* code combination which implements the protected logical process infringes on a patent for the process.[19] Patenting this underlying process would thus prevent others from avoiding infringement merely by changing individual instruction steps, or by substituting groups of instructions that perform the same function using different individual code steps.[20] However, this

16. This type of protection is somewhat analogous to the scope of protection obtained through the copyright laws. That is, under both schemes of protection, it is the object code that is protected. *See* Note, *Copyright Protection Of Computer Program Object Code*, 96 HARV. L. REV. 1722 (1983).

17. *See id.* at 1733-39. *See also* Note, *The Patentability of Computer Programs: Merrill Lynch's Patent For a Financial Services System*, 59 IND. L.J. 633, 637 (1985).

18. Samuelson, *supra* note 2, at 1026 n.2; Note, *The Disclosure Requirements of 35 U.S.C. § 112 and Software-Related Patent Applications: Debugging the System*, 18 CONN. L. REV. 855 n.1 (1986).

19. Note, *supra* note 17, at 637:

> [W]hile a competitor cannot use one's copyrighted program without permission, there is no prohibition [in copyright law] against devising a different program to perform the same functions. Patents are sought in order to provide [this additional] protection for the underlying idea. . . .

20. *See, e.g.*, Anthony, Jr. & Colwell, *Litigating The Validity and Infringement of Software Patents*, 41 WASH. & LEE L. REV. 1307 (1984).

> A patent on a computer program need not be limited to the manner of expression used by the programmer. Rather, the patent may protect the broad con-

level of protection raises a host of patentability problems.

1. The Mathematical Algorithm exception.

It has been argued that computer program algorithms are unpatentable because they are merely abstract mathematical algorithms.[21] This assertion, however, cuts too broadly. While computer program algorithms which are merely mathematical algorithms should not be protected by the patent laws, it is not the case that all such computer program algorithms are by definition mathematical algorithms as well.

Commentators and courts have struggled to find a principled basis for distinguishing "computer program algorithms" from "mathematical algorithms."[22]

The Supreme Court has long held that laws of nature, natural phenomena, and abstract ideas are excluded from patent protection.[23] For example, the Court has stated "a new mineral discovered in the earth or a new plant found in the wild is not patentable subject matter. Likewise, Einstein could not patent his celebrated law E=mc2; nor could Newton have patented the law of gravity. Such discoveries are 'manifestations of . . . nature, free to all men and reserved exclusively to none.' "[24]

In the early 1970s, the Court included among these "long established principles" patents which sought to protect mathematical formulae. Interestingly, the inventions in these cases, and those which followed, utilized digital computers as elements of the invention.[25] However, the courts avoided rejecting the claims at issue

---

cepts or ideas embodied in the program. Patent protection is typically expressed in broad terms which describe the overall organization or function of a computer program.

Id. at 1307-1308 (footnote omitted).

21. Samuelson, *supra* note 2, at 1123-25.

22. *See, e.g.*, Note, *supra* note 1, at 156 ("confusion about the meaning of the term 'algorithm' has been a major reason why the Supreme Court has refused to extend patent protection to software programs"); Note, *supra* note 17, at 633 ("[l]ower courts and patent practitioners are now struggling to determine how these cases fit into the general theory of patent law"); Note, *The Status of Patent Law Concerning Computer Programs: The Proper Form For Legal Protection*, 33 DRAKE L. REV. 155, 158-59 (1984), and sources cited therein; Comment, *Softright: A Legislative Solution to the Problem of Users' and Producers' Rights in Computer Software*, 44 LA. L. REV. 1413 (1984) ("[t]he flurry of literature and complaints from all sides indicate that there is a serious problem and that there is widespread dissatisfaction with the current state of the law").

23. *See, e.g.*, Parker v. Flook, 437 U.S. 584 (1978); Gottschalk v. Benson, 409 U.S. 63, 67 (1972); Funk Bros. Seed Co. v. Kalo Inoculant Co., 333 U.S. 127, 130 (1948); Rubber-Tip Pencil Co. v. Howard, 87 U.S. 498, 507 (1874).

24. Diamond v. Chakrabarty, 447 U.S. 303, 309 (1980).

25. Gottschalk v. Benson, 409 U.S. 63, 65 (1972), involved a method of converting

based on the use of such computers, directing their objections instead to perceived mathematical algorithms which underlay the invention. For example, in *Gottschalk v. Benson*,[26] the Supreme Court found that claims directed to an algorithm used to convert binary code decimal numbers to equivalent pure binary numbers fall outside of the scope of § 101. The court defined "algorithm" as a "procedure for solving a given type of mathematical problem" and concluded that such an algorithm is like a law of nature and is, therefore, nonstatutory.

Similarly, the claims at issue in *Parker v. Flook*[27] provided for a method for programming a digital computer with a mathematical control equation to update alarm limits during the catalytic chemical conversion of hydrocarbons.[28] The Supreme Court found the claims nonstatutory, characterizing the invention as seeking to protect a formula for computing an "alarm limit" from other known variables, without offering any new methods for selecting the variables to be used.[29]

In these cases, the Supreme Court, and the C.C.P.A. in subsequent cases, avoided framing the computer as the central element on which the question of patentability turned.[30] Thus, otherwise statutory subject matter is not rendered unpatentable merely because it contains an algorithm or recites a step to be performed by a computer. In *Benson*, the Court specifically stated that its holding did not preclude the granting of a patent "for any program servicing a computer."[31] In *Flook*, the Court stated that "a process is not unpatentable simply because it contains a law of nature or a mathematical algorithm."[32] The Court addressed this issue directly in *Diamond v. Diehr*.[33] There, the Court held that a process of curing synthetic rubber which included a step of constantly recalculating in a digital computer the appropriate cure time according to a

---

coded numbers to a system more easily understood by digital computers. Parker v. Flook, 437 U.S. 584, 586 (1978), concerned a method for programming a digital computer to update alarm limits during a hydrocarbon conversion process. *See infra* text accompanying notes 27-29.

26. 409 U.S. 63 (1972).

27. 437 U.S. 584 (1978).

28. *Id.* at 586.

29. 437 U.S. at 591-92.

30. For example, the Court stated in *Benson*: "It is said that the decision precludes a patent for any program servicing a computer. We do not so hold." 409 U.S. at 71.

31. *Id.*

32. 437 U.S. at 590.

33. 450 U.S. 175 (1981).

well-known mathematical equation, did fall within the scope of patentable subject matter defined in § 101.

> [R]espondents here do not seek to patent a mathematical formula. Instead, they seek patent protection for a process of curing synthetic rubber. Their process admittedly employs a well-known mathematical equation, but they do not seek to preempt the use of that equation. Rather, they seek only to foreclose from others the use of that equation in conjunction with all of the other steps in their claimed process.[34]

a. The C.C.P.A.'s two-step test of patentability.

In the period following *Benson* and *Flook*, the C.C.P.A. developed a working test to implement the Supreme Court's broad commands. The court limited the scope of *Benson* and *Flook* to claims principally or exclusively directed to *mathematical* algorithms as defined in *Benson*. In *In re Freeman*,[35] *In re Walter*,[36] and later, *In re Abele*,[37] the C.C.P.A. developed a two-pronged test to determine whether such claims fell within the definition of statutory subject matter. Under this test, a claim must be analyzed as a whole to determine 1) whether it recites, directly or indirectly, a mathematical algorithm;[38] and, if it does, 2) whether the algorithm is applied in any manner to physical elements or process steps.[39] An invention satisfies the requirements of § 101 either if it fails the first test, or if it satisfies both. That is, a claim that does not recite a mathematical algorithm falls within the scope of § 101, as does a claim in which a mathematical algorithm is somehow applied to physical elements or process steps.

The Supreme Court implicitly affirmed this two-part test in the case of *Diamond v. Diehr*.[40] Building upon the analysis of *Diehr*, the C.C.P.A. in *Abele* analyzed the second prong of the *Freeman-Walter* test as inquiring whether the claims, when viewed without the algorithm, contained otherwise statutory subject matter.[41]

---

34. *Id.* at 187.
35. 573 F.2d 1237 (C.C.P.A. 1978).
36. 618 F.2d 758 (C.C.P.A. 1980).
37. 684 F.2d 902 (C.C.P.A. 1982).
38. *Id.* at 905.
39. *Id.* at 908-09. *See also Walter*, 618 F.2d at 767.
40. The Court stated, "We recognize, of course, that when a claim recites a mathematical formula . . . an inquiry must be made into whether the claim is seeking patent protection for that formula in the abstract." 450 U.S. at 191. The Court goes on to say that "when a claim containing a mathematical formula implements or applies that formula in a [patentable] structure or process . . . then the claim satisfies the requirements of § 101." *Id.* at 192.
41. 684 F.2d at 909.

Thus, for the C.C.P.A., a computer program algorithm may be patentable, unless it would result directly or indirectly in the patenting of a mathematical algorithm, law of nature, or other otherwise unpatentable subject matter.

The mathematical/nonmathematical distinction has been criticized.[42] Much of the criticism and confusion arises from the Supreme Court's tautological definition of the terms "mathematical" and "algorithm." The Court in *Benson* defined what it called a "mathematical algorithm" as "a procedure for solving a given type of mathematical problem."[43] Of course, this definition merely begs the question of what is meant by "mathematical."

At some level, all computer programs are "mathematical" insofar as a computer operates internally by performing binary arithmetic operations. The C.C.P.A. has repeatedly avoided approaching the problem at this level, however. "This statement . . . would suffice to remove all computer-arts inventions from the scope of § 101. It is itself misleading because it ignores what the computer is doing, concentrating instead on how it is being done."[44] Thus, while every computer process is in some sense a "mathematical" algorithm, not all such programs are thereby rendered nonstatutory under the *Benson* exception to § 101.

The Supreme Court's definition has been criticized on another level. Professors Newell[45] and Samuelson[46] assert that all algorithms are "mathematical" as that term is defined by mathematicians. "Computer scientists . . . understand 'mathematics' to be a field concerned with defining abstract relationships among concepts and with defining rules about how those concepts should be manipulated."[47] Professor Newell puts it even more broadly: "Humans think by means of algorithms. Sequences of mental steps and algorithms are the same thing. . . ."[48] Professor Samuelson asserts that patents at the computer program algorithm level could be infringed by thinking through its steps or by practicing it in eve-

42. *See, e.g.*, Chisum, *supra* note 3, at 977-78; Newell, *supra* note 3, at 1024; Samuelson, *supra* note 2, at nn.296-345 and accompanying text.

43. 409 U.S. at 65.

44. *In re* Pardo, 684 F.2d 912, 916 (C.C.P.A. 1982); *In re* Walter, 618 F.2d 758, 769 (C.C.P.A. 1980); *In re* Application of Bradley, 600 F.2d 807, 812 (C.C.P.A. 1979), *aff'd by an equally divided Court*, Diamond v. Bradley, 450 U.S. 381 (1981).

45. *See* Newell, *supra* note 3, at 1024.

46. *See* Samuelson, *supra* note 2, at 1123.

47. Samuelson, *supra* note 2, at 1123.

48. Newell, *supra* note 3, at 1024-25. *See also* Note, *supra* note 1, at 172-73 ("[p]roblem solvers have not confined their use of algorithms to mathematical contexts").

ryday life.[49]

The problem with this criticism is that it is overly broad. That the subject matter of a computer program patent parallels the human thought process, does not necessarily render it unpatentable; all process patents that result from human invention bear the marks of the human thought process in that an inventor at some point envisioned the steps of the process in his or her mind.[50] Broadly speaking, any process, such as a chemical process, can be modelled as an algorithm plus its physical components.[51] Thus, a process patent for such a chemical process covers not only the particular physical combinations of chemicals, but also the underlying algorithm for combining them;[52] nonetheless, such processes are not thereby rendered nonstatutory.

For Professor Samuelson, the patentable difference lies in the "instantiation" of the underlying algorithm. "Instantiation" is defined as the embodiment of the inventive concept.[53] She presents an analogy: algorithms for making steel have their primary instantiation in the processing of raw materials in a steel plant, and is thus statutory, while in contrast, a computer program algorithm has its primary instantiation in computer programs written to implement them.[54] "Patentability has traditionally been judged by the nature of the primary instantiation anticipated by the procedure. . . . Mathematics has traditionally been considered part of the 'liberal arts.' "[55] Coupled with her assertion that "all algorithms are mathematical," she then concludes that all computer program algorithms are thereby nonstatutory.

It is not clear, however, that Professor Samuelson's analogy is precisely correct. One could just as easily describe a computer program algorithm as having its primary instantiation in a computer that has been loaded with a program, or in a digital system that has been permanently designed to execute the algorithm (i.e., one in which the machine code is embodied in a ROM or other permanent storage device).[56] In the same manner that the underlying al-

49. Samuelson, *supra* note 2, at 1124.

50. Disregarding, of course, those inventions which result from serendipitous discoveries.

51. *I.e.*, it can be described by a flowchart.

52. Indeed, the protection would be quite limited if it only covered the physical elements of the process.

53. Samuelson, *supra* note 2, at 1112 n.341.

54. Samuelson, *supra* note 2, at 1112.

55. *Id.*

56. In this case, it would not be precisely correct to say that the algorithm, or even the machine code, is "stored" as such in the computer. In actuality, a computer memory is a

gorithm for a process of making steel is "instantiated" in a particular industrial context, a computer program algorithm is designed for a particular purpose and is tied into a physical computer system (which can be quite elaborate depending upon the complexity of the program). The computer program receives input from electrical or mechanical input devices, manipulates electromagnetic signals, and perhaps operating attached magnetic storage media, display devices, etc. Neither the steel-making algorithm nor the computer program algorithm could be infringed just by thinking about it; the claims describing both processes can only be infringed by using the algorithms within their associated physical contexts.

The C.C.P.A. has defined the term "mathematical algorithm" narrowly. That is, the first prong of the *Freeman-Walter-Abele* test is satisfied if the process steps perform the function of carrying out a mathematical equation, i.e., if the steps operate arithmetically on numerical values to obtain other numerical values as an end product. In *In re Bradley*, affirmed by an equally divided Supreme Court,[57] the C.C.P.A. stated

> [i]t is true that a modern digital computer manipulates data, usually in binary form, by performing mathematical operations. . . . But this is only how the computer does what it does. Of importance is the significance of the data and their manipulation in the real world, i.e., what the computer is doing. It may represent the solution of the Pythagorean theorem, or a complex vector equation describing the behavior of a rocket in flight, in which case the computer is performing a mathematical algorithm and solving an equation. This is what is involved in *Benson* and *Flook*. On the other hand, it may be that the data and the manipulations performed thereon by the computer, when viewed on a human level, represent the contents of a page of the Milwaukee telephone directory, or the text of a court opinion retrieved by a computerized law service. Such information is utterly devoid of mathematical significance.[58]

The conflict between these two viewpoints is basically a difference in perspective. One position approaches the problem from a theoretical position, taking the view that *all* algorithms, broadly de-

---

physical array of digital switches that bears no resemblance to a sequence of process steps that we commonly call a program. *See* discussion *infra* at notes 95-109 and accompanying text.

57. 600 F.2d 807 (C.C.P.A. 1979), *aff'd by an equally divided Court,* Diamond v. Bradley, 450 U.S. 381 (1981).

58. *Id.* at 812.

fined, are mathematical.[59] The C.C.P.A., on the other hand, grounds its view firmly in the "real world" in which the term "mathematical" is defined "on a human level."[60]

Of the two viewpoints, surely the C.C.P.A.'s is the more preferable approach. Though it may be the less intellectually correct position, it does have the virtue of recognizing that inventors live and function in a world where fine analytic distinctions are often more confusing than necessary. Popular perception surely has a place in the patent laws.[61] Even though the patent laws seek to encourage inventive activity, an inventor may discover that his or her invention cannot be patented. One who has composed a novel and useful computer program will find little solace in the fact that René Descartes was able to abstract Euclid's theorems to representations,[62] or that those of greater than "ordinary skill in the art" deem it unpatentable. The "popular perception" model is particularly preferable when we consider that the determination of patent validity will ultimately be entrusted to politically appointed judges and juries who are unlikely to grasp fine mathematical distinctions.[63]

There may be situations where a computer program is in effect equivalent to a mathematical algorithm, as where the program is designed merely to execute a particular equation. In that case, the program would be nonstatutory. However, generally speaking, a computer program is not *per se* a "mathematical algorithm" simply because it is a computer program. This is so even if the program incorporates in its logic sections or subroutines which execute mathematical equations.[64] Claims directed to a computer program should be presumptively viewed as containing otherwise statutory

59. Professor Samuelson concedes that her definition runs counter to "popular perception." Samuelson, *supra* note 2, at 1123.

60. *Bradley*, 600 F.2d at 812.

61. For example, 35 U.S.C. § 103 codifies the standard of obviousness as that of a person of "ordinary" skill in the art, a term which can be roughly analogized to the ubiquitous "reasonable man" standard in negligence cases. Panduit Corp v. Dennison Mfg. Co., 810 F.2d 1561, 1566 (Fed. Cir. 1987), *cert. den.*, 481 U.S. 1052 (1987) ("person having ordinary skill . . . not unlike the reasonable man"). Both tests measure the conduct at issue against popular expectations of propriety in a particular situation.

62. *See* Samuelson, *supra* note 2 at 1123 n.390.

63. This may also be construed as an argument in favor of removing computer programs altogether from the scope of the patent laws. The ultimate question here, however, is where to draw the line — at a theoretical point, or at a point defined by "popular perception." This author merely asserts that the latter is preferable to the former.

64. In that case, the second prong of the *Freeman-Walter-Abele* test would be implicated. *See* discussion *infra* at notes 67-89 and accompanying text.

subject matter unless they fall within another judicial exception to § 101.

Defining the term "mathematical algorithm" in this way, and thereby limiting the scope of the judicial exception, is analytically sound.[65] A computer program which merely rephrases an abstract mathematical formula or equation in computer code is nothing more than the equation itself, phrased a different way, and should not be patentable. On the other hand, defining the terms broadly, as Professors Newell and Samuelson would define them, would run counter to the Supreme Court's express disclaimer that its decision was not intended to preclude the patenting of any program servicing a computer.[66]

b. The second prong of the C.C.P.A.'s test.

The C.C.P.A. has also adopted this liberal approach in its interpretation of the second prong of the *Freeman-Walter-Abele* test. Under this test, a claim which has been found to recite a mathematical algorithm is nevertheless statutory if the algorithm is applied in any manner to physical elements or process steps.[67] Essentially, the test inquires whether a mathematical algorithm or formula is the principal or exclusive subject matter of the claim, or whether the claim merely utilizes the equation in a particular context.[68]

The Supreme Court painted with a broad brush in the rationale of the rejection of the claims in *Diehr*: "The sole practical application of the algorithm was in connection with the programming of a general purpose digital computer."[69] It further stated that "[t]ransformation and reduction of an article 'to a different state or thing' is the clue to the patentability of a process claim that does not include particular machines."[70] Taken in isolation, these statements may be understood as precluding the patenting of precisely the sort of computer programs discussed above, programs designed for use in a computer which stands alone and which is not part of a larger process. This section will explore the implications of the

65. The policy reasons in support of this interpretation are discussed *infra* at note 94 and accompanying text.

66. *Benson*, 409 U.S. at 71.

67. *See supra* note 39 and accompanying text.

68. *Abele*, 684 F.2d at 906 (subject matter of claim not patentable "if it would wholly preempt an algorithm . . . or if it would preempt the algorithm but for limiting its use to a particular technological environment").

69. Diamond v. Diehr, 450 U.S. at 185-86.

70. *Id.* at 184.

"transformation" requirement of *Diehr* on the patentability of computer programs as such.

Processes which do involve the transformation of a physical thing into a different state or thing are traditionally protectable by patents. Over a century ago, the Supreme Court stated, with reference to the Patent Act of 1793,

> A process is a mode of treatment of certain materials to produce a given result. It is an act, or a series of acts, performed upon the subject matter to be transformed and reduced to a different state or thing. If new and useful, it is just as patentable as a piece of machinery. In the language of the patent law, it is an art. The machinery pointed out as suitable to perform the process may or may not be new or patentable; whilst the process itself may be altogether new, and produce an entirely new result. The process requires that certain things should be done with certain substances, and in a certain order; but the tools to be used in doing this may be of secondary consequence.[71]

The computer in *Diehr* was merely one element in an industrial process for curing synthetic rubber which clearly transformed a physical thing into something different. The Court took pains to describe the process as one which was otherwise protectable under the patent laws.[72] In fact, the Court took the peculiar position of viewing the mathematical algorithm and computer as potential *bars* to the patenting of an *otherwise* statutory invention (a position also taken in *Benson* and *Flook*).[73]

In *Flook*, the Supreme Court framed the issue as "whether the identification of a limited category of useful, though conventional, post-solution applications of such a formula makes respondent's method eligible for patent protection," and answered that question in the negative.[74] Similarly, the Court cautioned in *Diehr* that "insignificant post-solution activity will not transform an unpatentable principle into a patentable process," but held the activity there to be statutory.[75] Thus, novel or significant post-solution activity may well make such a claim patentable, at least where such activity serves to satisfy the second prong of the *Freeman-Walter-Abele* test.

Where the inventor's contribution to the art lies in a novel logical process or flowchart, it is difficult to see why patent protection

71. Cochrane v. Deener, 94 U.S. 780, 787-788 (1877).

72. *See, e.g., Diehr*, 450 U.S. at 184, 187.

73. *See id.* Professor Samuelson points this out in her very thorough discussion of *Diehr*. Samuelson, *supra* note 2, at 1094-99.

74. 437 U.S. at 585.

75. 450 U.S. at 191-92.

should turn on whether the process is tied to a particular mechanical process. An abstract logical sequence which is unattached to a particular application may be unpatentable as a purely "mental process."[76] For example, an applicant should not be able to patent the process of sorting any set of numbers, or to patent the idea of a nested loop. On the other hand, many programs are by their nature restricted to particular applications or contexts. It is true that programs linked with particular mechanical or industrial processes are task-specific; however, programs designed for specific purposes, such as defining a particular word processing program, or a particular video game, are often equally so. Protection of these programs in and of themselves will not remove basic ideas from the public domain nor unduly preempt implementations designed for other applications. Though it is not always easy to categorize a particular program as either "abstract" or application-specific — indeed, these two definitions merely define the ends of a spectrum, rather than two pigeonholes — requiring a "physical transformation" or "post-solution activity" is particularly inappropriate as a defining test.

The artificiality of this test is easily demonstrated. For example, suppose two inventors design identical computer programs for use in known industrial processes. Inventor A discloses that the program is to be used in the particular industrial process, but claims only the program logic. Inventor B, on the other hand, explicitly claims the logic plus the known steps of the industrial process. If a "transformation" or other "post-solution activity" were required for a computer program to be patentable, Inventor B would be entitled to a patent, but Inventor A would not be entitled to a patent for developing the identical program.[77] This would be true even if the other steps of the industrial process belonged to the public domain and B could not obtain a patent on the process without the program. It would also be true if Inventor A's program were clearly inapplicable to any context except the process in question (and hence the argument that his program would preempt other uses fails). The distinction in this case would seem to be more artificial than anything else — B's invention is patentable over A's simply because well-known physical steps were explicitly included in the claims, even though those steps were implicitly equally necessary to A's invention.

Or take a second example. Inventor A has designed a novel

76. *See* discussion *infra* at notes 90-93 and accompanying text.

77. Of course, we will for this essay ignore the issues of interferences, infringement, and double-patenting.

word processing program, which can be loaded into a home computer. He claims the logic of the program as his invention. Inventor B designs an electronic word processing terminal (which operates identically to A's invention), and claims the terminal and digital circuitry as his invention. A general purpose computer, when loaded with A's program, is physically identical to the digital configuration of B's digital circuitry,[78] but only temporarily so. Under this reasoning, Inventor B would be entitled to the patent, but Inventor A would not be, even though their inventive acts were identical.

The courts seem to have recognized the artificiality of this distinction, and have responded by liberally interpreting the definition of "transformation" and "physical" state or thing. For example, in *Freeman*, the C.C.P.A. dodged the question entirely. In addressing the Patent and Trademark Office's (PTO) contention that providing a "fleeting display on a cathode ray tube" as the solution to the algorithm was insufficient post-solution activity under *Flook*, the court stated that since there was no mathematical algorithm involved in the invention, as there was in *Flook*, there was no question of "post-solution activity" in that case.[79] Though the court sidestepped the issue, this statement, or more precisely, what the court did not say, has potentially broad implications. It is apparent from this decision that a "pure" computer program which operates on input data and whose only end result is a display on a cathode ray tube (CRT), is not *per se* nonstatutory under *Benson* and *Flook*.

In *In re Johnson*,[80] *In re Sherwood*,[81] and *In re Taner*,[82] the C.C.P.A. addressed a trilogy of applications directed to seismic detection through wave propagation. *Johnson* was directed to a method for removing unwanted noise components from seismic wave data to form noise-free seismic traces.[83] *Sherwood* was directed to converting amplitude-versus-time seismic traces into amplitude-versus-depth seismic traces.[84] *Taner* claimed a method of simulating substantially planar or substantially cylindrical seismic

78. *See, e.g., In re* Bernhart, 417 F.2d 1395, 1400 (C.C.P.A. 1969) ("if a machine is programmed in a certain new and unobvious way, it is physically different from the machine without that program; its memory elements are differently arranged"). *See also* discussion *infra* at notes 96-109 and accompanying text.

79. 573 F.2d at 1246.

80. 589 F.2d 1070 (C.C.P.A. 1978).

81. 613 F.2d 809 (C.C.P.A. 1980), *cert. den.*, 450 U.S. 994 (1981).

82. 681 F.2d 787 (C.C.P.A. 1982).

83. 589 F.2d at 1070.

84. 613 F.2d at 811.

energy waves from substantially spherical seismic waves.[85] In all three, sequences of electrical signals were formed from reflected energy waves to produce different sequences of electrical signals. Though the court stated that all three methods claimed mathematical algorithms, and thus satisfied the first prong of *Freeman-Walter-Abele*, it also found that all three inventions satisfied the second prong, and thus were statutory. "[T]he claims were, as a whole, drawn not to a method of solving that algorithm but to a process of converting one physical thing into another physical thing."[86] "Seismic traces are . . . physical apparitions."[87] "That those 'physical apparitions' may be expressed in mathematical terms is in our view irrelevant."[88]

Thus, speaking in the context of mathematical algorithms, the C.C.P.A. has broadly defined the "transformation" requirement to include the manipulation of electrical signals.[89] It would seem, then, that implementing a computer program algorithm in a digital computer would satisfy this requirement. After all, the acts of receiving electronic data signals, internally manipulating them into other signals, and outputting new signals, are precisely what a computer does.

2. The Mental Steps exception to § 101.

Closely related to the mathematical algorithm exception is the doctrine that mental steps cannot be patented as such. Indeed, these two doctrines derive from the same policy that abstract ideas are not patentable.[90]

The Mental Steps doctrine is particularly important in the context of computer program patentability. At some level, all programs are "mental steps" in that they represent a logical process. This is, however, no reason to deny patent protection to all such programs as a fixed rule.

---

85. 681 F.2d at 787.

86. *Id.* at 790.

87. *Sherwood*, 613 F.2d at 813.

88. *Taner*, 681 F.2d at 790.

89. The C.C.P.A. dispensed with this requirement entirely in such cases as *In re* Toma, 575 F.2d 872 (C.C.P.A 1978), *In re* Deutsch, 553 F.2d 689 (C.C.P.A. 1977), *In re* Musgrave, 431 F.2d 882 (C.C.P.A. 1970), *In re* Bernhart, 417 F.2d 1395 (C.C.P.A. 1969). In those cases, the claims were held statutory even though the processes in question arguably would not have been statutory but for the fact that they were performed by a computer.

90. The PTO has implemented this policy in its requirement that to be patentable, an idea must be reduced to practice, not merely conceived.

The MANUAL OF PATENT EXAMINING PROCEDURE[91] at § 2106 sets out an illustrative example of a bare set of source or object code instructions. The illustration there seeks to protect a set of instructions for comparing two arrays to generate a third array. The MPEP states that the set of instructions is probably not protectable "when not associated with a computing machine to accomplish a specific purpose", because it represents a mere idea or abstract intellectual concept of a programmer.[92] This is a correct conclusion. Where the computer program algorithm is nothing more than a description of an abstract process written in computer code, it is nonstatutory. Not all computer program algorithms, however, are necessarily abstract. Further, a computer program algorithm may well implement an abstract idea as one step or element (for example by utilizing the MPEP's example code in a subroutine in a larger program), and nonetheless be statutory subject matter.

To use a different example, one cannot patent the idea of a nested loop, because it is a mere abstraction. Though the C.C.P.A.'s two-part test is not directly applicable to this situation because no mathematical algorithm is implicated, the rationale for the second prong of the test applies equally in this context. The idea of a nested loop, like a mathematical equation, is not patentable in itself; it is only patentable when applied in a particular context.

The Supreme Court recognized this distinction when it stated that "it is now commonplace that an *application* of a law of nature or mathematical formula to a known structure or process may well be deserving of patent protection."[93] Thus, a computer program

---

91. U.S. DEP'T OF COMMERCE, MANUAL OF PATENT EXAMINING PROCEDURE § 2106 (5th ed. 6th rev. 1987) [hereinafter MPEP].

92. *Id.* at 2100-4. The example given is as follows:

"consider the following claim[ ]:

(2) 'A computer program for comparing an array A(N) with array B(M) to generate array C comprising the steps of:

```
Do 70 N=1, 10
  Do 80 M=1,20
    If A(N) = B(N) then C(M) = B(M)
      80 Continue
    70 Continue * * *' "
```

The MPEP then asserts that "[t]his bare set of instructions fails to recite subject matter that falls within any statutory category. In this regard, a bare set of computer instructions does not set forth a sequence of steps which could be viewed as a statutory process. Such a computer language listing of instructions, when not associated with a computing machine to accomplish a specific purpose, would not constitute a machine implemented process, but would constitute nonstatutory subject matter as the mere idea or abstract intellectual concept of a programmer, or as a collection of printed matter." *Id.*

93. *Diehr*, 450 U.S. at 187 (emphasis in original). *See also* Mackay Radio & Tel. Co. v.

which implements a mental idea in a larger context may be patentable as the application of an underlying algorithm or idea to a specific context. Of course, a program algorithm which is nothing more than the computer code version of an otherwise unpatentable abstract idea does not thereby become statutory merely because it is implemented on a computer.[94] In that situation, the mental idea could not be said to have been applied in any manner to a process.

For example, though the nested loop described in the MPEP may not be patentable in itself, the code or flowchart implementing a particular program, such as a word processing program, which includes such a loop could be considered patentable as an application of the formula to a particular process. In this example, the known structure or process would be the particular computer system in which the program is stored and executed. The computer would be a known structure applied to a particular purpose — for example, as a word processor or video game console.

3. Policy justifications.

The crux of an unpatentable mental idea or mathematical algorithm is that it is unlimited in application, and yet cannot be applied to any utilitarian function on its own. Thus, the law of gravity broadly describes a physical phenomenon that may be applied to many different contexts, from ballistic missiles to golf club designs, and yet one cannot "use" the principle as such. One can only create an invention, perhaps patentable, that takes advantage of the manifestation of the principle in a particular way. The principle itself can be applied in different contexts to create diverse inventions. In contrast, a patentable invention is an application of such a principle, useable in and of itself (possibly with the aid of other known inventions or applications) for the specific utilitarian purpose for which it was designed.

Described this way, it is clear that a computer program algorithm is more than just a mental idea or law of nature. Such program algorithms are designed for particular utilitarian purposes. A program drafted as a video game cannot normally be applied to a different computer to form a word processing program. It can,

---

Radio Corp. of America, 306 U.S. 86, 94 (1939), *reh'g den.*, 306 U.S. 608 (1939) ("[w]hile a scientific truth, or the mathematical expression of it, is not a patentable invention, a novel and useful structure created with the aid of knowledge of scientific truth may be").

94. As the C.C.P.A. stated, "Claims to nonstatutory processes do not automatically and invariably become patentable upon incorporation of reference to apparatus." *In re* de Castelet, 562 F.2d 1236, 1244 (C.C.P.A. 1977).

however, be used in and of itself with the aid of a known digital computer system.

It is important to note that defining a "program" as the underlying logical process conforms fully with the fundamental purposes of the patent laws, that disclosure and patenting of an invention should add to, and not detract from, the store of knowledge in the public domain. The rationale for granting protection for applications of ideas is simply that by disclosing to the public a novel, useful and nonobvious *application* of a principle otherwise "free to all men and reserved exclusively to none,"[95] an inventor has advanced the useful or technological arts for the benefit of society, and is rewarded with a patent grant. This rationale applies equally to computer programs. A nested loop is unpatentable; however, novel, useful and nonobvious applications or implementations of such a loop contribute to the public store of knowledge.

At this definitional level, a "program" is necessarily limited in its context and utility, so that patent protection would not preempt abstract ideas or laws of nature otherwise available to all. A "program" under this definition is a specific logical sequence designed for implementation on a particular computer system to achieve particular purposes (such as executing a word processing program or a video game on an IBM-PC-compatible system). Granting such protection would not preempt the adaptation of similar logic, or even the copying of portions of the patented code, for different purposes. Nor would such protection preclude the use of dissimilar methods to perform the same purpose. In fact, such programs are often much more limited in scope and application than are analogous mechanical or industrial processes. This is particularly true given the vastly different types of computer systems currently in use. For example, a program sequence designed to prioritize multiple requests in a sixty-four bit multiprogram environment would be completely inapplicable to most eight-bit personal computer systems.

C. *Patenting a program as a machine part.*

A third possible approach under which a computer program may be patentable lies in considering a computer program, when stored in a computer's memory, as a physical part of a machine. Under this view, a computer program may be patented as a physical apparatus, rather than as a process.

It is important to note that technologically, though an al-

95. *Diehr*, 450 U.S. at 185.

gorithm may be a logical sequence of steps, it is not stored as such in a computer memory. Rather, when loaded with a program, a computer memory is a spatial array of digital elements (electromagnetic signals) which operate electronically. Because a single step of source code may represent several steps of machine code, and thus many individual digital operations, a computer does not necessarily execute an algorithm in a fashion which parallels the appearance of the flowchart or code on paper.[96] Programs which appear nearly identical at even the object code level may bear little resemblance to each other when their respective arrangements in memory are compared.

The C.C.P.A. recognized this distinction early on, when it stated that "if a machine is programmed in a certain new and unobvious way, it is physically different from the machine without that program; its memory elements are differently arranged."[97] Because a computer stores program commands as binary elements, this protection would seem to be necessarily limited to the protection of particular object code as it is stored in a particularly configured computer.[98]

The C.C.P.A. held in *Bradley* that a so-called "firmware" module, hardware elements permanently programmed with a microcode, is statutory within § 101.[99] The court stated that

> [A]ppellants have characterized their combination of hardware elements as a mechanism which enables the computer to alter information in its system base. . . . They are in no way claiming the altered information; in fact, the particular information acted upon by appellants' invention is irrelevant to the operation of the invention itself.[100]

This is undoubtedly a correct result.[101] Such single-purpose computers — digital circuitry which has been permanently configured to perform a certain process, such as an electronic calculator, the chip in a digital watch, or a Texas Instruments "Speak &

---

96. *See, e.g.*, Von Spakovsky, Von Spakovsky & Graffeo, *The Limited Patenting of Computer Programs: A Proposed Statutory Approach*, 16 CUMB. L. REV. 27, 38-39 (1985) (discussing the relationship between source and object code).

97. *In re* Bernhart, 417 F.2d 1395, 1400 (C.C.P.A. 1969).

98. Different computers have different memory storage and addressing techniques; hence, this protection would, if narrowly defined, seem limited to particular computer systems.

99. *In re* Bradley, 600 F.2d 807 (C.C.P.A. 1979), *aff'd by an equally divided Court*, Diamond v. Bradley, 450 U.S. 381 (1981).

100. *Id.* at 812-13.

101. For a different view, see Samuelson, *supra* note 2, at 1047 & n.91 and at 1123 & nn.389-394.

Spell" — have long been considered statutory subject matter under § 101 independently of the data which the devices are designed to manipulate. Indeed, an electronic calculator is simply the solid-state electronic equivalent of a conventional adding machine. On this point, the C.C.P.A.'s language in *Bradley* is revealing:

> We see no difference . . . with respect to being within § 101, between appellants' claimed invention and a strictly mechanical adding machine, which is certainly statutory if claimed in a manner which does not embrace any particular calculation that the machine is capable of making.[102]

Thus, speaking broadly, computer "hardware" that has been configured for a particular purpose is clearly patentable.

The *Bradley* court expressly cautioned that "[i]f appellants were claiming the information embodied in the firmware or the firmware itself, *per se*, a different case would be presented. We express no opinion on the statutory nature of such an invention. . . ."[103] The court emphasized several times that it addressed only the "combination of hardware elements, one of which happens to be . . . microprogrammed in a particular manner."[104]

In so limiting its decision, the C.C.P.A. left open the question of whether a general purpose computer programmed in a particular way could be patented as a sort of "temporary" firmware. That is, a general purpose computer, when programmed in a particular fashion, could be thought of as a single-purpose computer (like a digital calculator), at least as long as it is so programmed. The computer program on which protection is actually sought would then be analogous to the microcode used to program the *Bradley* firmware module.

Protecting the program in this manner would remove many of the patentability problems that arise when trying to patent the logical sequence underlying the program. However, other analytical questions would be raised.

1. Scope of protection.

The first question would be the scope of protection actually obtained under this method. As briefly described above,[105] because of the differences in the ways computers utilize their memory addresses, this form of protection would be limited to the protection of

102. *Bradley*, 600 F.2d at 812.
103. *Id.* at 813.
104. *Id.*
105. *See supra* note 97 and accompanying text.

object code within a particular computer system. This may well be sufficient; in reality, most prepackaged computer programs sold commercially are indeed object code packages tailored for specific computer systems.

2. Enablement problems.

35 U.S.C. § 112 requires that a patent application "shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise and exact terms as to enable any person skilled in the art to which it pertains . . . to make and use the same. . . ."[106] This so-called "enablement" requirement may be satisfied by claiming a particular product or substance as the result obtained by carrying out a specified process.[107] When framed as a product-by-process claim, the patented invention is still the resulting product itself. That is, the making, using or selling of that product would infringe the patent even where a different process is used to produce it.

An interesting conceptual question is whether the protection obtained could be expanded by claiming the programmed computer hardware in the form of a product-by-process claim. More specifically, the enablement requirement may be met by simply disclosing in the patent the computer program, or even the flowchart of the program logic, and claiming the resulting programmed computer as the hardware obtained as a result of the disclosed programming process. This question has potentially important ramifications, because permitting a program to be patented in this way could theoretically expand the scope of protection from the particular object code as stored in the memory of a particular computer, to protect the broader underlying logical process (flowchart) that the object code is designed to implement. That is, in addition to protecting the object code, an inventor could conceivably foreclose others from using the underlying logical process or computer program algorithm, if using that program algorithm produces an equivalent object code.[108]

Even if no such additional protection could be obtained, draft-

106. See Note, *supra* note 18, at 856 n.12, for a discussion of enablement in the context of computer programs.

107. *See* MPEP, § 706.03 (e).

108. A patent entitles the holder to foreclose others from making, using or selling the patented invention or its "equivalents." *See supra* note 6 and accompanying text. An "equivalent" has been defined to be subject matter which performs "substantially the same function in substantially the same way to achieve the same result" as does the patented subject matter. Sanitary Refrigerator Co. v. Winters, 280 U.S. 30 (1929).

ing the patent as a product-by-process invention may nonetheless be the preferred method of claiming the programmed computer. Because of the § 112 enablement requirements, it may be difficult and lengthy to describe adequately the precise configuration of electromagnetic signals which particularly characterizes a computer programmed in a specific way. Due to the complexity and level of detail at which modern computers operate, even simple computer programs would become extremely complex if the inventor is required to pinpoint and identify each and every memory location. Indeed, a product-by-process claim may be the only feasible way of accomplishing the dual interests of enablement and brevity.

3. Equivalents.

A third question is whether a programmed computer, when viewed in this manner, would infringe upon analogous hardware or firmware patents. For example, would a patent for a general purpose computer when programmed in such a way that it receives inputs from a numerical keyboard and performs mathematical computations thereon, be considered to infringe upon a patent for a digital calculator, or even upon a mechanical adding machine? Put another way, would a random-access memory (RAM) be equivalent for patenting purposes to a similarly configured read-only memory (ROM) unit.[109]

Intuitively, it would seem that a programmed computer that does precisely the same thing as a ROM chip should, at least in some instances, not be patentable as a separate invention. For example, coin-operated video arcade games are typically programmed with ROMs. Permitting a competitor to then produce a home-computer version of the same video game without infringing a patent on the arcade version would seem to render that patent effectively useless.[110] On the other hand, it would seem equally ridiculous to interpret the scope of such a patent so broadly that a general purpose computer programmed in a particular way would be infringed by a completely mechanical device. For example, it would push the patent laws beyond their purpose to permit a home computer tempo-

109. In this context, the term RAM, or Random Access Memory, is used to describe an electronic memory unit which only stores its contents temporarily, automatically erasing its contents when commanded to do so or when the computer is turned off. A ROM, or Read-Only Memory, on the other hand, refers to a memory unit whose contents are stored permanently without change.

110. In this case, production of the home-computer version may be barred under applicable copyright, trademark or trade dress laws.

rarily programmed to perform arithmetic operations to be infringed by a digital calculator or mechanical adding machine.

These are difficult conceptual questions in themselves, and they are beyond the scope of this article. They are raised here only to point out the difficulties that need to be addressed before the patenting of computer programs will be an effective and desirable form of protection. In the author's view, however, as discussed above, that these difficult questions exist is no reason in itself to completely deny patent protection to computer programs.

IV. THE PTO'S POSITION

During the 1960s-1970s, the PTO strongly opposed the patenting of computer-program related inventions. Between 1969 and 1982, the C.C.P.A. reversed the PTO's denial of patent on computer-program related inventions a total of eighteen times.[111]

Diamond v. Diehr apparently marked a change of heart for the PTO. Since the *Diehr* decision was announced in 1982, the C.C.P.A./C.A.F.C. reversed the PTO only four times.[112]

The PTO has formally announced its new position with respect

111. *In re* Prater, 415 F.2d 1393 (C.C.P.A. 1969); *In re* Bernhart, 417 F.2d 1395 (C.C.P.A. 1969); *In re* Musgrave, 431 F.2d 882 (C.C.P.A. 1970); *In re* Benson, 441 F.2d 682 (C.C.P.A. 1971), *rev'd sub nom.* Gottschalk v. Benson, 409 U.S. 63 (1972); *In re* Knowlton, 481 F.2d 1357 (C.C.P.A. 1973); *In re* Comstock, 481 F.2d 905 (C.C.P.A. 1973); *In re* Johnston, 502 F.2d 765 (C.C.P.A. 1974), *rev'd sub nom.* Dann v. Johnston, 425 U.S. 219 (1976); *In re* Noll, 545 F.2d 141 (C.C.P.A. 1976), *cert den.*, 434 U.S. 875 (1977); *In re* Chatfield, 545 F.2d 152 (C.C.P.A. 1976), *cert den.*, 434 U.S. 875 (1977); *In re* Deutsch, 553 F.2d 689 (C.C.P.A. 1977); *In re* Waldbaum, 559 F.2d 611 (C.C.P.A. 1977); *In re* Flook, 559 F.2d 21 (C.C.P.A. 1977), *rev'd sub nom.* Parker v. Flook, 437 U.S. 584 (1978); *In re* Freeman, 573 F.2d 1237 (C.C.P.A. 1978); *In re* Toma, 575 F.2d 872 (C.C.P.A. 1978); *In re* Johnson, 589 F.2d 1070 (C.C.P.A. 1978); *In re* Application of Bradley, 600 F.2d 807 (C.C.P.A. 1979), *aff'd by an equally divided Court*, Diamond v. Bradley, 450 U.S. 381 (1981); *In re* Application of Phillips, 608 F.2d 879 (C.C.P.A. 1979); *In re* Application of Sherwood, 613 F.2d 809 (C.C.P.A. 1980), *cert. den.*, Diamond v. Sherwood, 450 U.S. 994 (1981).

The C.C.P.A. also affirmed the denial of patent for computer-program related inventions in *In re* Christensen, 478 F.2d 1392 (C.C.P.A. 1973); *In re* Sarkar, 588 F.2d 1330 (C.C.P.A. 1979); *In re* Application of Gelnovatch, 595 F.2d 32 (C.C.P.A. 1979); *In re* Application of Maucorps, 609 F.2d 481 (C.C.P.A. 1979); *In re* Application of Walter, 618 F.2d 758 (C.C.P.A. 1980).

For an excellent discussion of these cases, see Chandler, *Proprietary Protection of Computer Software*, 11 U. BALT. L. REV. 195, 230-255 (1982).

112. *In re* Taner, 681 F.2d 787 (C.C.P.A. 1982); *In re* Abele, 684 F.2d 902 (C.C.P.A. 1982); *In re* Pardo, 684 F.2d 912 (C.C.P.A. 1982); *In re* Iwahashi, 888 F.2d 1370 (Fed. Cir. 1989). The court sustained the PTO's rejections in *In re* Meyer, 688 F.2d 789 (C.C.P.A. 1982) and in *In re* Grams, 888 F.2d 835 (Fed. Cir. 1989). Additionally, the PTO Board of Appeals and Interferences affirmed the examiner's rejection in *Ex parte* Murray, 9 U.S.P.Q. 2d (BNA) 1819 (Bd. Pat. App. & Inter 1988), which was not appealed to the Court of Appeals for the Federal Circuit.

to computer-related inventions in two published opinions.[113] This section will explore the PTO's current practice as reflected in these published opinions and guidelines.[114]

A. *The Manual of Patent Examining Procedure.*

In 1987, the PTO issued the latest revision of its Manual of Patent Examining Procedure (MPEP). Section 2106 of this revision, entitled "Patentable Subject Matter — Mathematical Algorithms or Computer Programs," sets forth broad guidelines by which computer-program patent applications are to be considered.

It is interesting to note that the MPEP sharply distinguishes between inventions involving mathematical algorithms and those involving computer programs. The MPEP discusses *Diehr*, *Bradley*, and *Flook* only in the context of claims which "directly or indirectly recite[ ] a mathematical algorithm."[115] The manual then recites in separate paragraphs the procedure for "analyzing computer program related claims."[116]

The Manual first states broadly that "computer implemented 'processes are encompassed within 35 U.S.C. § 101 under the same principles as other machine implemented processes, subject to judicially determined exceptions . . . *In re Johnson*.' "[117] The Manual then goes on to describe the two-prong *Freeman-Walter-Abele* test:

> In accordance with the two-step procedure outlined above, claims seeking coverage for a computer program would be non-statutory . . . only if, when considered as a whole, they merely recite a mathematical algorithm, or a method of calculation which is not applied in any manner to physical elements or process steps.[118]

The manual then sets forth the hypothetical program loop described above[119] as the only concrete example of a "judicially determined exception[ ] outside the mathematics area."[120]

---

113. *See* MPEP, § 2106; *Patentable Subject Matter*, 1106 Official Gazette U.S. Pat. & Trademark Off. OG5 (1989).

114. These published opinions do not have the force of law. The author's interviews of patent examiners, however, have indicated that these publications, coupled with the case law, provide the primary (sometimes exclusive) guidance to examiners in their examination of computer program-related patent applications.

115. MPEP, § 2106, at 2100-3.

116. *Id.* at 2100-4.

117. *Id.*

118. *Id.*

119. *See supra*, note 91 and accompanying text.

120. MPEP, *supra* note 112, at 2100-4.

B. *The statement of the Solicitor.*

The actual guidelines presented in the MPEP are brief and provide little real guidance. The PTO provided a more detailed analysis in an opinion published in 1989 and endorsed by the Solicitor.[121]

The PTO again sharply distinguished between claims involving mathematical algorithms and those involving computer programs.[122]

Section III of the opinion begins by attempting to define what is meant by "computer program." "A 'process' or 'algorithm' is a step-by-step procedure to arrive at a given result. In the patent area, a 'computer process' or 'computer algorithm' is a process, i.e., a series of steps, which is performed by a computer. A '[computer] program is a sequence of coded instructions for a digital computer.' *Benson*."[123] The article then observes that

> [b]oth the series of steps performed by a computer, and the software directing those steps, have acquired the name 'computer programs'. . . . What is sought to be protected by patent is the underlying process.[124]

Thus, the PTO has adopted the view that computer programs are to be defined as the logical processes that underlie the computer code sequences.

The PTO opinion then opens its analysis by asserting that "[t]he Supreme Court has not ruled on whether computer processes are *per se* statutory or nonstatutory. The decisions in *Benson*, *Flook*, and *Diehr* all dealt with claims viewed as mathematical algorithms."[125] The opinion describes *Pardo* as holding that computer processes are statutory unless they fall within a judicially determined exception. "The major (and perhaps only) exception in the area of computer processes is the mathematical algorithm."[126] The opinion then reasons that "[i]f a computer process claim does not contain a mathematical algorithm in the *Benson* sense, the second step of the *Freeman-Walter-Abele* test is not reached, and the

121. *Patentable Subject Matter, supra* note 112. The opinion is described as a "recent legal analysis . . . on the subject of the patentability of mathematical algorithms and computer programs. The analysis is published for the benefit of the public." *Id.* at 5.

122. The analysis is presented in three sections. Section I sets forth the general requirements of 35 U.S.C. § 101. Section II is entitled "Mathematical Algorithms." Section III is entitled "Computer Programs."

123. *Patentable Subject Matter, supra* note 112, at 10-11.

124. *Id.* at 11.

125. *Id.* at 11.

126. *Id.*

claimed subject matter will usually be statutory."[127] The opinion then goes on to cite several C.C.P.A. decisions to support its conclusion that such processes are patentable.[128]

Finally, the PTO opinion asserts that

> [a]rguably, other exceptions such as 'methods of doing business' and 'mental steps' may be raised if a claim is not a true computer process, but merely recites that an otherwise nonstatutory process is performed on a computer. . . . These would appear to be exceptions with very narrow application to claims which are not limited to implementation by a machine.[129]

C. *Conclusions concerning the PTO's approach.*

The MPEP and the Solicitor's analysis provide little real insight or guidance into the PTO's practice in examining computer program patent applications. The two publications are little more than broad endorsements of certain C.C.P.A. decisions. Arguably, the publications go even farther than do the cases which they cite.[130]

Though the actual approach taken by the PTO examiners in the examination of such application may be difficult to discern from these publications, it is clear that the PTO has broadly and wholeheartedly embraced the position that computer programs, defined as the underlying logical processes, are as a general rule included within 35 U.S.C. § 101. It is interesting to note that the skilled engineers and scientists in the PTO have soundly rejected the proposition that all computer programs are necessarily mathematical algorithms. Indeed, perhaps no single fact is more clear from the PTO's published opinions.

V. CONCLUSION

Computer programs hold a unique place in our society. It is beyond dispute that the development of novel computer programs has had a profound effect on technology. But because of their *sui generis* nature, they are not easily amenable to protection under the

127. *Id.*

128. *See id.*

129. *Id.*

130. The Solicitor's analysis does not even attempt a scholarly, objective examination of the case law; instead it selectively extracts dicta which supports its ultimate conclusion of patentability. Interestingly, the analysis cites with approval the Delaware District Court's decision in Paine, Webber, Jackson & Curtis, Inc. v. Merrill Lynch, Pierce, Fenner & Smith, Inc., 564 F. Supp. 1358 (D. Del. 1983), a decision which is singled out for criticism by Professor Samuelson. *See* Samuelson, *supra* note 2, at 1120-1122.

patent laws. This is no reason, however, to deny patents where the program is novel, useful and nonobvious. It is all too easy a solution to simply propose that computer programs should be precluded entirely from receiving patent protection. Instead of searching for justifications for denying such programs the benefits of patent protection, we should instead be searching for ways to incorporate them into the patent system.